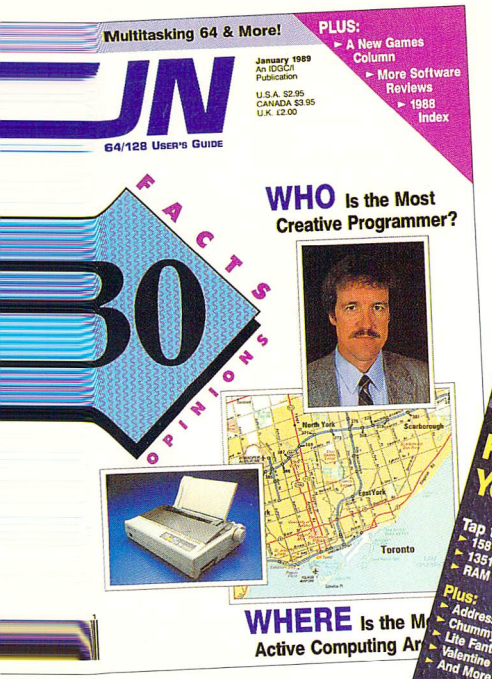
January/February 1989 Edition

RE RUN

RUN Programs on Disk

For the C-64 and C-128



s: Extra Bonus Program!

Introduction

January–February '89 ReRUN

HAPPY NEW YEAR! My New Year's resolution is far easier to state than to accomplish. My 1989 goal is to provide ReRUN users with a record-breaking number of useful and interesting programs for both the C-64 and C-128. With some software manufacturers moving away from the Commodore 8-bit line, it is crucial for publications such as ReRUN to meet the resulting needs in the areas of productivity, utilities and entertainment.

Let's begin with a calendar-making application, named 2001: A Calendar Program. This C-64 program allows you to make up to 12 appointments on any given date and will save them to disk for future reference. As its name implies, it'll keep records into the 21st century.

Do you use an Amiga or other multi-tasking computer? If so, you'll fully appreciate the work performed by college professor Michael Ingrassia, who created MOS (Multitasking Operating System) 1.0 for the C-64. MOS 1.0 reconfigures the C-64's operating system to allow you to run up to 3 programs at once on your C-64. Demo programs are included on the disk to demonstrate this amazing feat.

One of the C-128 programs in this edition of ReRUN is Raising Rainbows, a challenging—albeit exasperating—exercise in growing what has to be the world's fussiest and most sensitive houseplant.

On a completely different theme, Screen Basic, for the C-64, makes simple work of programming video screens by adding 13 commands to Basic 2.0.

February programs begin with Address, Please?, another program written exclusively for our C-128 audience. Using either a 1351 mouse or a joystick, you can flip through the alphabetized pages in your 40-Column mode address book, much like a traditional Rolodex.

One of my favorite C-64 games of late, Chummy Checkers, is also included on this disk. Before I began playing this computerized version of checkers some time back, I ranked right up there with some of the worst checker players in the history of the game. By playing against the computer and analyzing its moves, I found my game immensely improved.

Programmers and programmers-to-be are certain to find the

Mouse and RAM Expander programs by Tom Brown to be invaluable additions to their program collections. Not only are programs included, but the article is packed with useful information on using these devices.

Trace the Light Fantastic brings your computer to life with animated lines that draw designs in the direction determined by moving the joystick left or right. These lines are drawn in Hi-Res mode, resulting in sharp, clearly defined lines that really draw attention to the screen.

February is a good month to remember U.S. presidents, Valentine's Day and a former computer magazine editor. While it has nothing to do with Presidents' Day, A Love-ly Idea brings a Valentine spirit to your computer and the work of a former RUN Associate Editor back into our ReRUN series. Jim Borden, author of many popular RUN programs from 1985 through 1987, wrote this handy C-64 application. A Love-ly Idea lets you design simple greeting cards on the screen and print them out.

Finally, our bonus program, Multicolor Editor 64 (MC-Editor), transforms your computer into a graphics workstation, complete with utilities for creating, saving and loading user-defined graphics images to disk. As proof of the versatility of MC-Editor, we've included an impressively animated building-construction scene on this disk.

In past years, our January–February edition of ReRUN has included a file of the Index of *RUN*'s programs and articles published during the past year. This edition is no exception: You'll find the 1988 Index in the form of a sequential file and in RUN Script format. Just load it into RUN Script or Easy Script and print it out.

As with every edition of ReRUN, this is my least favorite part of the introduction—the time I have to say, "That's all, folks."

Technical Editor
RUN *magazine*

How To Load

Loading from Menu

To get started, C-64 users should type LOAD "MENU 64",8 and press the return key. When you get the Ready prompt, the menu is loaded and you should type RUN to see a list of the programs on your disk. C-128 users need only press the shift and run-stop keys. When all the programs are displayed on the screen, you can run the one you select by pressing a single key.

Loading from Keyboard

If you do not wish to use the menu program, follow these instructions.

C-64: To load a C-64 program written in Basic, type: LOAD "DISK FILENAME",8 and then press the return key. The drive will whir while the screen prints LOADING and then READY, with a flashing cursor beneath. Type RUN and press the return key. The program will then start running. To load a C-64 program written in machine language (ML), type: LOAD "DISK FILENAME",8,1

C-128: All C-64 programs can be run on the C-128 as long as your computer is in C-64 mode. All C-128 programs are clearly labeled on the directory page. Your C-128 *must* be in C-128 mode to run these programs. To load a C-128 mode program, press the F2 key, type the disk filename and then press the return key. When the program has loaded, type RUN.

Making Copies of ReRUN Files

Many programs on your ReRUN disk have routines that require a separate disk onto which the program writes or saves subfiles. To use these programs, you must first make a copy of the original program onto another disk that has enough free space on it to hold these newly written subfiles.

It's simple to make a copy of a Basic program. Just load it into your computer as outlined above, and then save the program back onto a separate disk that has plenty of free space for extra files.

Copying an ML program is not so simple. You cannot simply load and save an ML program; you'll need to use a disk-backup utility program, such as the one on your Commodore Test Demo disk.

Directory

Page	Documentation	Disk Filename	File Type
		*MENU 128	BASIC
		MENU 64	BASIC
1	2001: A Calendar Program	2001:CALENDAR	BASIC
3	Multitasking Comes To the C-64	MOS BOOT	BASIC
		MOS VARIABLES	BASIC
		MOS CLOCK	BASIC
		MOS.HEX	BASIC
		+ MOS ML	ML
7	Raising Rainbows	*RAISING RAINBOWS	BASIC
8	Screen Basic	SCREEN BASIC 64	BASIC
		SB LOADER	BASIC
		DEMO SCR.BASIC	BASIC
		SB +	ML
		TEST.SCN	ML
12	Address, Please?	*ADDRESS BOOK.128	BASIC
		*AD. DATA	BASIC
14	Chummy Checkers	CHUMMY CHECKERS	BASIC
15	Program Your mouse And RAM Expander	*EXPNDER BUG-FIX	BASIC
		RAM EXPNDER 64	BASIC
21	Trace the Light Fantastic	TRACE THE LIGHT	BASIC
		LIGHT LOADER	BASIC
22	A Love-ly Idea	A LOVE-LY IDEA	BASIC
25	Multicolor Editor 64	£MC-EDITOR.0	ML
		SC1.1	ML
		SC1.2	ML
		SC1.3	ML
		SC1.4	ML
		SC1.5	ML
		SC1.6	ML
		SC1.7	ML
		SC1.8	ML
		CS1	ML
		SC3	ML
		SC4	ML
		CH2	ML
		CH4	ML

* — C-128 mode only
£ — Bonus program

Before you run a program, carefully read the documentation that pertains to it.

RUN it right: **C-64; printer optional**

2001: A Calendar Program

By Ken Huebner

LIFE IN THE FAST LANE is a hallmark of the twentieth century, and, as we approach the year 2000, the pace is getting even faster. If you're having trouble remembering special events, appointments and deadlines now, how will you cope in the years to come? Well, you should do just fine with the help of Calendar 2001, the twenty-first century calendar program for your twenty-first century lifestyle.

Written in Basic, Calendar 2001 uses relative files for instant reads and writes to its data files. Variable MT at the beginning of the program has a default value of 12, which represents the maximum number of reminders you expect to input for each day. Do not change the value of MT once you've run the program and created a month file.

If you have a 1541 disk drive, 12 reminders per day will fill a disk in about one year; with a 1571, 25 per day will fill the disk in about the same period. Larger values will require scratching old reminders to make room for new ones from time to time, or will necessitate the use of a new reminder disk.

Once MT is set, you may want to save a backup to disk. Then format a fresh disk to hold your reminders. You could also save a copy of the program to your data disk, so everything's together and easier to use; just be sure there's a backup elsewhere.

To use Calendar 2001, load it using MENU 64, insert the data disk and type in RUN. When the program asks "What's the date?", enter today's date in numerical form, with the numbers separated by commas (12,20,1988), and press the return key (it is not necessary to type in leading zeroes with one-digit numbers).

Soon a menu will appear with the date you input at the top. There are six menu options:

1.— See the Calendar
2.— Read from Calendar
3.— Make an Appointment
4.— Save a Note

5.— Erase from Calendar
6.— Exit to Basic.

Option 1 displays a calendar showing the days of the current month. Press the cursor-up key to move forward to the next month or the cursor-down key to move back a month. When you've finished viewing the calendar, press return.

Option 2 lets you read a selected day's reminders. Just enter the date, again in numerical form, and the list will appear on-screen. On the left are numbers you'll need for identifying your reminders when using other options. On the right are the date, and perhaps a time, for each reminder, along with the contents of the reminder.

You can review the list any number of times by pressing the space bar. You can also make a hard copy by just pressing P. Press the return key when you're finished.

Option 3 is for creating an appointment reminder and saving it to disk. At the query, enter the date of the appointment to load that day's file into the computer. This load will take longer when you're first using the program or starting a new month, because the program has to open a new file first. Once the file is available, Calendar 2001 will ask "Appointment with whom?" Reply with a name that's no more than 20 characters long. Then, at the queries for time, enter a number from 1 to 12 for the hour and a number from 0 to 59 for the minutes.

You also need to indicate whether you want to be reminded of the appointment in advance. If not, answer no or press return, and the reminder will be saved only to the date of the appointment. If you answer yes, you'll be asked how many days ahead of time you want the reminders to start. Enter a number from 1 to 14, and the reminder will be saved to the actual date, plus the specified number of days before.

Option 4 is for creating other types of reminders and saving them to disk. Except that there's no query for time, this option works just like option 3.

Option 5 erases reminders. Type in the date concerned, and, when the reminders appear on-screen, locate the one you want to erase, note its I.D. number and press return. Then, at the query, type in the I.D. number and press return again to erase the reminder from your data file.

When you're finished using Calendar 2001, press 6 to exit to Basic.

This program is most effective if used daily, so try to get into that habit. Then you'll be on a clear course into the 21st century!

Multitasking Comes To the C-64

By Michael Ingrassia

SO, YOU THOUGHT your C-64 could never do multitasking! Well, think again. Now that capability of its big brother, the Amiga, is yours with MOS (multitasking operating system) 1.0, which lets you edit and run two or three Basic programs at the same time.

Possible applications of MOS are myriad and challenge the imagination. How about playing background music with one program while playing a game with another or printing a clock on one screen to keep track of your time running a program on another?

Maybe you'd like to display a document file while checking the code that accompanies it; compare two versions of a program by running them alternately; examine a disk directory without destroying the program in memory; or display variable values while a program is running? All are within easy reach once you start using MOS.

And even if you never use MOS for multitasking, its ability to load three programs at once so you can run or edit them in any order without accessing the disk drive makes it worthwhile to have on every disk you use.

To explain multitasking, here's an analogy from college life. If, as frequently happens, two adjunct professors share a desk, at the end of the first professor's office hours, he gathers up his materials from the desktop and puts them into his briefcase, leaving the desktop free for his colleague's use. Then the second professor comes in, removes the materials he needs from his briefcase and sets them out on the desk. Changing the contents of the desktop constitutes a context switch.

My desk, on the other hand, is single-tasking. I don't share it with anyone, and it's covered with all sorts of things—books, correspondence, a nameplate, my wedding picture. I couldn't possibly pop all those things into a briefcase to give someone else a place to work.

The Basic operating system that's built into the C-64 is like me. It "takes over the desk," getting out its wedding picture and nameplate and generally making itself at home in memory. Or, to be more exact, it fills memory with data, values and addresses of routines needed to run a program. To make Basic multitasking, so context switches can take place between Basic programs in the C-64's memory, these values and addresses must be saved and restored at each switch. That's what MOS does.

Program Overview

The C-64 provides only 38K of Basic workspace, but most Basic programs occupy only a fraction of that, so workspace is not a major limitation for MOS. (Divide the number of blocks a program fills by 4 to determine roughly how many K it fills. Most of my Basic programs are under 7K in length.) MOS chops the 38K into three areas called "spreads," plus an area for its own workings. Spread 1 can hold a 10K program, spread 2 a 12K program and spread 3 an 11K program.

To run three programs at once without becoming hopelessly confused, you need more than one screen. The Amiga, for example, "partitions" the screen into individual windows for the programs that need to display output. However, this would be difficult to accomplish on the C-64, so I've used "window shades" instead of windows. Pull the top shade all the way down and that's all you see. Raise it halfway and pull the second shade all the way down, and you see half of the first shade and half of the second. Lift both shade screens. and the third screen appears as a backdrop. (See below for information on the keys to manipulate these shaded areas.) The top shade shows spread 1's output, the second shade shows spread 2's output and the stationary backdrop shows spread 3's output.

You're probably wondering how the CPU can execute three programs at once. Well, of course, it can't really, but by dividing its processing time equally among the three programs, it gives that illusion. This division of effort is called time-slicing, because each program gets slices of the CPU's time in "round-robin" fashion. If one of the programs finishes executing before the others, the CPU just shifts to dividing its time between the two still running.

You'll notice the time-slicing, because each of your programs will run about one-third slower than normal. Actually, it will run even a little slower than that, due to the complexity of context-switching. Still, MOS is cheaper than buying three computers!

Normally, only the current screen is affected by input and output, including input from the keyboard. Whichever program gets to the keyboard buffer first gets any input from the keyboard, so make sure you never have two programs wanting input at the same time. In Interactive mode, there's no problem, because you can specify which spread gets the input by using the Spread command (see below).

Using MOS

"BOOT MOS" is the main MOS program. Run it, and portions of all three screens will appear, with green for the top shade, orange for the middle shade and purple for the backdrop. The cursor will be blinking on the green screen, indicating that you're ready to begin working with spread 1.

Use the following function keys to raise and lower the shades:

F1—Lowers the top shade.
F2—Raises the top shade.
F3—Lowers the second shade.
F4—Raises the second shade.

These keys operate in both Direct and Program modes, and they're also placed in the keyboard buffer for use by your programs. If you wish, you may write your programs in such a way as to ignore the function key input or disable shade-moving before your program executes by entering POKE 38226,0. To restore use of the function keys, enter POKE 38226,255. For now, hold down F1 until the top shade has descended completely.

To witness three programs running simultaneously, load each into a spread as follows:

```
LOAD "MOS CLOCK",8
SPREAD 2
LOAD "MOS VARIABLES",8
SPREAD 3
LOAD "MOS VARIABLES",8
```

Next, type SRUN <return>, and all three will execute. This sequence assumes you're starting with spread 1, but you can change the order if you want. Also, if you're running only two programs, there's no need to load anything into an empty spread.

MOS Commands

SPREAD n (n *must* have the value 1, 2 or 3) lets you switch from one spread (spread 1, by default) to another.

SRUN actually runs the programs simultaneously. You can be in any spread when you enter this command. To stop one program from running, tap the run-stop key as usual, but to stop all the programs, you may need to hold the key down. Immediate mode is locked out until you get the message SRUN FINISHED.

To run a single program, just go to the desired spread and type RUN; SRUN is unnecessary.

Note that you may accidentally execute a SPREAD or an SRUN if you have a syntax error in a command with READ or RUN in it. That's how MOS characterizes the new commands.

Miscellaneous Notes

You'll notice that the separate screens have steady raster edges until you perform a disk operation. Because of the flickering during disk accesses, I recommend that you have only one screen visible at those times.

The background color of the spreads 1, 2 and 3 can be changed with a POKE 38213, POKE 38214 or POKE 38215, respectively. Avoid poking 53281, because you can't tell which screen will change color.

Should you accidentally press run-stop/restore while using MOS, reinitialize it by entering

```
SPREAD 1 <return>
SYS 37891 <return>
```

whether you can see your input on the screen or not.

You may experience some difficulties with program compatibility when using MOS. More than one program, including MOS, may need the same locations in memory. Also, the spread programs may want to divert some of the low-memory vectors that MOS has diverted for its own purposes.

Programs (even machine language programs!) that stick to Basic-style memory management are most likely to run with no problem. Your best bet is to run only programs that you thoroughly understand. And keep in mind that MOS doesn't use any of the 4K of memory from location 49152 up.

RUN it right: **C-128**

Raising Rainbows

By Mary Wilson

ARE YOU TIRED of crawling in dreadful dungeons and fighting off incredible creatures? Tired of zapping aliens, gobbling dots and frantic races against the clock? Relax with this slow-paced, nonviolent game that will test your skill at windowsill horticulture. Here's the scenario:

You've just returned from an expedition to the Amazon jungle, where you happily stumbled upon a few specimens of the rare and jealously guarded rainbow plant, so named because a single plant produces flowers of various colors. After wading through miles of red tape, the government finally rewarded you for locating the elusive plant by giving you permission to take one small specimen out of the country.

Now that you have your plant safely at home, your top priority is to carefully nurture it and coax it to bloom, so you can gather seeds to start more plants to sell. Success in this venture will assure your fortune, as flower growers around the world will pay fabulous sums for even two or three seeds. So, grab your watering can and box of fertilizer, and let's see if your thumb is green.

How Much Water?

Since you're a horticulturist, it's assumed that you've potted your plant in the proper soil with the proper drainage. Now you must maintain a delicate balance of light, water, fertilizer and pot size. Your first decision is in which window to place the plant. After you choose, a window will be drawn, with your plant on the sill and the plant's condition printed in red in the lower-right corner of the screen. At the lower left you'll see a menu of things the plant might need. You can select one or just leave the plant alone for a while.

As your rainbow plant grows, more decisions will be necessary. For example, it will occasionally need a larger pot so it doesn't get rootbound and die. Be careful, though: If you choose a pot that's too large, the plant won't be able to absorb the water in so much

soil and will, in effect, die from overwatering. If the plant does die, the program will tell you why, so you can avoid the same situation the next time.

You can move your plant from window to window to get the best exposure. As you do so, the color of the curtains will change to help you remember from which direction the light is coming.

You may notice some unusual screen characters in the curtains. These are standard keyboard graphics, but some of them look strange because of the distortion that occurs when printing to a hi-res screen. In this case, the distortion is an advantage.

If all its needs are met, your plant will grow ten leaves, and then flower stalks and buds will appear. Finally, if it's still in good condition, it will burst into bloom. Your C-128 screen will be full of color, and your thumb will be green!

RUN it right: **C-64**

Screen Basic

By John Ryan

NORMALLY, VIDEO PROGRAMMING on the C-64 entails a myriad of Peeks and Pokes to accomplish the simplest task, and multiple-screen processing can be a daunting challenge to even the most seasoned programmer. Screen Basic alleviates this problem by adding 13 video programming commands to Basic 2.0. These commands provide an easy-to-use programming environment that even includes multiple screens and raster interrupts.

Screen Basic runs in both Program and, to a limited degree, Direct modes. It features three separate video screens, each with semi-independent color RAM, as well as separate background and border colors, and, by setting up raster interrupts, you can display two of the screens simultaneously. The program also provides a command to copy the Commodore character set from ROM into bank 0 or 2. This, used with the Raster command, will display two full custom character sets at once. You can save, load, "grab" and "call" video screens with Screen Basic, as well as read the error channel of your

disk drive—without a line full of cryptic commands.

Load Screen Basic, using MENU 64. Then enter NEW and SYS 50176 to install it in the C-64's operating system.

THE COMMANDS

Below are explanations of all the new commands Screen Basic provides. Each command must be preceded by an @ symbol, or a syntax error will result. Also note that most of the commands require one or more parameters.

@VIDEO <screen number>. As I mentioned, Screen Basic provides three separate video screens. Video 1 starts at memory location 1024 ($0400), video 2 at 32768 ($8000) and video 3 at 33792 ($8400). These addresses must be used when poking information to the various screens. Print operations to the screens are handled automatically by the program.

Since there's only one memory area for color RAM information, each time you switch from one screen to another, the current color RAM is stored in memory before the switch is made. When you return to the first screen, the color is restored. This scheme ensures the integrity of your color display, regardless of the screen you're on. (See the @Type command, below, for special considerations.)

@HUE <screen number, color number>. The Hue command changes the background color of any video area. The colors are numbered 1–16, just as they're listed on the C-64 keyboard. If the screen specified is the current video area, the change is immediate; if not, it becomes apparent when you bank to the specified screen. (Screen Basic remembers the color of each screen.)

@BRDER <screen number, color number>. Use the Brder command to change the border of a screen to a specified color. The rules are the same as for the @Hue command.

@TYPE <screen number>. This handy command lets you print to any of the screens, regardless of which one is currently displayed. For example, by entering @TYPE 2 you can send information being printed to the video matrix starting at 32768, even if video 1 is on the screen.

Note that even though the characters are being printed to a different area of memory, color RAM for both video areas is the same. Whenever possible, you should do your printing in the same foreground color as your current screen, or use the @Raster command (below) to "hide" the color changes.

You can return to your current screen only by issuing another

@Type command. @Video won't work in this instance, since your original video area won't have changed.

@ERASE <screen number>. The Erase command clears a specified video area without clearing the color RAM of your current screen. It's a good idea to use this command during program initialization, as the video areas may be filled with random data. It works for all video areas, regardless of the one you're currently in.

@GRAB <screen number>. Grab stores the screen and color information for the current video area in memory. Three distinct storage areas are set aside for this purpose, so all three screens can be stored at once. Note that the screen number must be the same as your current video area for this command to work properly, and @Grab won't store hi-res screens. (Also see @Call, next.)

@CALL <source screen number, target screen number>. This versatile command lets you move previously grabbed screens to any video area. By calling two or more screens in succession, you can even achieve "page flipping" animation effects. Be careful of the syntax for this command: The source number is the number of the area previously grabbed, and the target number is the screen you're calling it to (your current screen, if color RAM preservation is important—this is not a swap command).

@RASTER <beginning scan line, ending scan line>. This powerful command displays two screens simultaneously. The two parameters, which can range from 1 to 254, tell the program where the division between the screens should be and where to restore the first screen. For instance, @RASTER 100,254 will display video area 1 up to scan line 100, then video area 2 to scan line 254. When the screen is divided, the background and border colors of both video areas are displayed at once.

Note that only scan lines 40–253 are visible, so anytime you use parameters 1–39 or 254, the interrupt will take place off the visible screen. Keep in mind, too, that this command can be used only while you're on video screen 1 or 2, because those are the screens the raster interrupt displays.

To achieve a rock-solid division, don't specify a scan line that will cut through the middle of printed screen characters. Also, if you wish to move sprites across the raster "seam," remember that two individual banks of memory are being displayed. In order to display a sprite in video bank 2, the sprite definition must also be stored in video memory from 33784, for sprite 1, to 33791, for sprite 8.

The formula for finding the sprite pointer is (storage ad-

dress – 32768)/64. Thus, a sprite image stored at 40960 ($A000) would have a pointer of 128—(40960 – 32768)/64. The best place to store sprite images in bank 2 is from 40960 to 45056, if a full alternate character set is not being used, or at 33792, if video screen 3 is not being used.

Don't reinitialize Screen Basic with the Raster command activated.

@OFF. This command turns off the raster interrupt.

@MEM <1 or 2>. The @Mem command makes it easy to use custom character sets. (In fact, it's the only way you can use custom character sets with Screen Basic.) If 1 is the parameter, the character set will be copied from ROM into RAM, starting in bank 0 at location 12288 ($3000). If the parameter is 2, the set will be copied into memory starting at 40960 ($A000). With an @Raster command, you can display both sets simultaneously. Just remember to protect the bank 0 character set from Basic by lowering or raising Basic memory, if your own program is very big.

To return to the standard character set, just reinitialize Screen Basic with SYS 50176.

@PUT <"filename">. The @Put command saves the video area you're viewing to a special disk file denoted by the suffix .scn. Both screen RAM and color RAM information is saved. Don't include the .scn in the command, because the program supplies it automatically. The device number is also supplied automatically. To change it from default 8, see Program Notes, below.

@SLD <"filename", screen number>. This screen loading command reads a previously saved screen file from disk. With @Sld, it's easy to load several files at once when a lot of information must be presented and color RAM preservation is important. The file can be read into any video area specified after the filename, regardless of which screen was saved with the @Put command. If you specify the video area you're currently viewing, you'll see the loaded screen immediately; if not, the screen and color information will appear when you bank to the video area the screen was loaded to. As with the @Put command, don't include the .scn filename suffix in an @Sld command.

@DISK. Use this to view the error status of the last device accessed.

Program Notes

Screen Basic occupies memory locations 50176–52162 ($C400–$CBC2) and uses locations 32768–50175, so both areas are unavailable for your use. The program protects these memory areas by

lowering the top of Basic to 32768, which chops about 8000 bytes off the top of free Basic memory.

You can change the disk drive device number for screen loads and saves by poking 1 into location 52126.

If you use the run-stop/restore key combination on any screen but video area 1, you won't be able to see what you're typing, and the computer will seem locked up. You can get out of this by entering @TYPE 1, thus returning the screen editor to video area 1; but, better yet, disable run-stop/restore with POKE 808, 239: POKE 792, 193.

You can't use variables as parameters for any of the Screen Basic commands. To include Screen Basic commands in a Basic program you're writing, place the following three-line routine at the beginning of your listing:

```
10 IF FLAG = 1 THEN 30
20 FLAG = 1:LOAD"SB + ",<device number>,1
30 SYS 50176
```

Due to the nature of video programming, I originally wrote Screen Basic to operate in Program mode only. However, after realizing that some of the commands (@Hue, @Brder) could be useful in Direct mode, I added a limited Direct Mode option. In this option, only one Screen Basic command is permitted per line, and it must be first. You can still have multiple Basic 2.0 commands in a line, as long as they follow the Screen Basic command. For example, @RASTER 1,100: PRINT "test" is legal, while PRINT "test":@HUE 1,1 and @RASTER 1,100:@HUE 1,1:@BRDER 1,1 are not. In Program mode, any combination of commands may be used.

RUN it right: **C-128 (in 40-Column mode); joystick or 1351 mouse**

Address, Please?

By Neil Hansen

ADDRESS BOOK 128 is an easy-to-use, graphics-oriented program that's operated with a joystick or a 1351 mouse, so there's no need to learn and type in commands. Just move the pointer on the screen

with the mouse or joystick and press the button to activate the program functions. Names and addresses are kept in alphabetical order, with room for up to 100 names under each letter of the alphabet, and the entries can be printed out on envelope labels.

PROGRAM FUNCTIONS

When you run Address Book 128, it first asks for your mode of input. Press 1 for a joystick and 2 for the 1351 mouse. After you make your selection, the screen goes blank for six seconds while the program is being set up. Note that the program takes full advantage of the mouse's Proportional mode.

The next display features a large rectangle on the left and a narrow rectangle on the right that contains the letters of the alphabet. Within the big rectangle, you see the following categories: Last Name, First, Middle, Address, State, Zip, Miscellaneous and Phone. At the bottom, there's an imitation page-fold and four boxes with the words Quit, Delete, Print and Save in them. The pointer rests in the center of the big rectangle, and the letter A appears in the upper-left corner, indicating that you're in the file for names beginning with A.

To enter the file for a different initial letter, first make sure a formatted work disk is in the drive. Then move the pointer to the alphabet rectangle, place it on the letter you want to load and press the button. The new letter replaces the A in the upper-left corner of the big rectangle, showing the file you are currently in.

When you want to enter information on a person, move the pointer to Last Name and start typing. When you're finished, press return.

You can flip through your address book by moving the pointer to the lower-left corner of the screen, where there's a small square divided into two triangles by a diagonal, making it appear as though the corner of the page had been folded up. To flip forward, move the pointer to the upper-right triangle and press the button; to flip backward, point on the lower-left triangle and press the button.

You can delete outdated cards by moving the pointer to the box that says Delete. But watch out! Deletion is permanent; you won't be able to get your card back.

To print out a card as an address label, make sure your printer is on and then move the pointer to the Print box.

If you want to save the names you've added to a letter file, move the pointer to the Save box and press the button. Be sure not to press another letter before you've saved your current additions, or the additions will be lost!

When you insert a new record into Address Book 128, you may encounter a small problem if you press the mouse button before it reaches the "Last Name" field. If the screen pointer turns red and will not move above any particular field, just press the return key, and the pointer will turn black. You can then use the mouse to position the cursor wherever you desire.

Leaving the program is as easy as moving the pointer to the Quit box. That's it—short and simple, but very handy!

RUN it right: **C-64; joystick**

Chummy Checkers

By Tony Brantner

BET YOU HAVEN'T played checkers for a long time—or maybe you've never even learned. My computer version is designed to change all that by breathing life into the old game.

If you're unfamiliar with checkers, the board is divided into eight rows of eight squares each—just like a chessboard. The two players must try to clear each other's pieces from the board or hem the pieces in so they can't be moved. In my version, you can play against another person or the computer, and you can also watch the computer play itself. When people play, a single joystick plugged into port 2 controls both sides.

When you start the game, you must first enter the number of people playing; then the player with the white pieces moves first.

Moving a piece involves two steps. First, use the joystick to position the flashing yellow cursor over the piece you want to move and press the firebutton. This makes the entire square flash. Next, move the cursor to the destination square and press the firebutton again. If the move is legal, the piece advances to the new square and your turn ends. If it's illegal, the coordinates clear so you can try again.

Jumping an opponent's piece to banish it from the board is done in the same manner, except your turn doesn't end automatically after you move. Instead, you have the option of jumping again by repeating the second step. When you want to end your turn, move

the cursor to any square but the one that's flashing and press the firebutton. After each turn, the number of pieces each player has left is displayed at the sides of the screen.

You can move your pieces only in a forward direction until they reach your opponent's back row. Then, when they are "kinged," they can move forward or backward. Kinged pieces are identified by a K in the center.

In the one-person version of Checkers, the computer controls the black pieces and you control the white. Although the computer plays defensively, it can't resist the chance to jump an opposing piece, so, by using one of your pieces as bait, you may be able to set up a multiple jump for yourself.

Entering zero when you're prompted for the number of players puts the computer in Auto-Play mode. This feature is handy for getting acquainted with the game or studying the computer's strategy.

To end play, press the F1 key. If you press F1 during the computer's turn, the machine will finish its move before the game ends.

RUN it right: **C-64 or C-128; 1351 mouse; 1700, 1750 or 1764 RAM expander**

Program Your Mouse And RAM Expander

By Tom Brown

TWO OF THE NEWEST hardware accessories for the C-64 and C-128 owner are the 1351 proportional mouse and the 1764, 1700 and 1750 RAM expanders from Commodore. While these devices open up new computing opportunities, learning to program them can be frustrating to experienced and novice programmers alike, since they don't work like other Commodore devices.

The Mouse

The first mouse for the C-64 and C-128, the Commodore 1350, operates similarly to a joystick (or an upside-down trackball). While

it has two buttons on top, only the left one can be used; the right button is wired in such a way that it can't be read properly. Since the 1350 is, in reality, nothing more than a joystick, its movements are often difficult to control.

The latest mouse from Commodore, the 1351, improves the situation by operating, not as a joystick, but as a pair of paddles. You may remember the paddle controller from the old Atari Pong-type games, where the only required movement is left-right or up-down. One end of the paddle is connected to a constant power source, the other to the SID (sound) chip, and between those two points there's a variable resistor (or potentiometer, from which we get the nickname "pot" for each paddle).

As you turn the paddle's knob, the voltage reaching the SID chip varies between 0 and +5, then the chip converts this voltage to a one-byte number in the range 0–255. Two paddles are normally connected to a single port, with one read at address 54297 ($D419) and the other at 54298 ($D41A), no matter which port is used.

It would be nice if we could use these two bytes as-is for screen coordinates. However, there are two problems. First, unless you're in Multicolor mode, the screen is more than 255 pixels (dots) wide. Second, the conversion process isn't steady; the value for each paddle "jitters" back and forth, making it necessary to read the paddle many times and calculate an average.

The 1351 avoids these problems by using a whole new procedure. It keeps track of its own movements, automatically relaying its current X-Y position to the computer via the two SID chip locations mentioned above—one location for X and the other for Y.

Now, remember that I said the value in each location is a one-byte number (in the range 0–255). However, only six of the eight bits in that byte are meaningful (the highest and lowest aren't used). In other words, the actual X and Y values transferred to the computer range from 0 to 63.

Although the values of both X and Y are now smaller than the range of available screen positions, they are reliable and free of that annoying jitter. To use these smaller numbers, the *last* value read from the mouse must be stored and compared to the current value. This is because, when either the X or Y value reaches 64, it wraps around to 0! Comparing old and new values tells which direction the mouse is moving, so it's apparent whether a current value of 0 is actually 0 or 64, 128 or some other number.

It's necessary to keep an eye on the mouse at all times, so the

movement is fast and accurate, but this can't be done from Basic. The only way the mouse can be used effectively is with a machine language routine included in the "housekeeping" chores the computer performs 60 times a second in its "interrupt" routine. Fortunately, the 1351 is packaged with a utility disk containing what we need, including several Basic programs that demonstrate how to use the machine language mouse reader for both the C-64 and C-128.

The Mouse Readers

The C-64 mouse reader is called M1351.64.BIN and loads into memory at 49152 ($C000). To use the mouse in the front port, activate the reader with SYS 49152; to use it in the rear port, activate it with SYS 49155. As you move the mouse around, the reader changes the position of sprite 0. Therefore, if your Basic program needs to know that position, use the following line:

```
X = PEEK(53248) + ((PEEK(53264)AND1)*256):Y = PEEK(53249)
```

Naturally, to see the mouse pointer move around the screen, you'll have to activate sprite 0. If you use the pointer definition (called Mouse.Pointer) on the 1351 disk, remember that it loads at 3584 ($0E00), which will interfere with your Basic program, unless the program is very short or you change the start-of-Basic pointer to point above 3584. The following lines activate the sprite, set its color to white, position it on the screen at 100,100 and set the sprite definition to 3584:

```
V = 53248:POKEV + 21,1:
POKEV + 39,1:
POKEV + 0,100:POKEV + 1,100:POKEV + 16,0:
POKE2040,56
```

The buttons on the 1351 mouse appear as joystick values, so reading them is relatively simple by using the following Basic line:

```
B = PEEK(C)AND17:B = ABS((B = 1) + (B = 16)*2 + (B = 17)*3)
```

Variable C should have a value of 56321 if you're using the front port or 56320 if you're using the rear port. The value returned in B is 0 if no button was pressed, 1 if the left button was pressed, 2 if the right button was pressed and 3 if both were pressed. This lets you use the value in On/GoTo commands in your own programs.

The C-128 reader is called M1351.128.BIN and loads at 6384 ($1800). To use the mouse in the front port, activate the reader with

SYS 6144; to use it in the rear port, activate it with SYS 6147. This reader operates the same way as the C-64 version—that is, by manipulating the position of sprite 0. However, Basic 7.0 provides commands that make it easier to use sprites:

```
X = RSPPOS(1,0):Y = RSPPOS(1,1):REM Find X and Y position
SPRITE 1,1,2:REM Turn sprite on
MOVSPR 1,100,100:REM Position sprite at 100,100
```

It's easier to use the pointer definition from the 1351 disk with the C-128, since the 128 reserves that memory location specifically for sprite definitions. The above sprite commands replace the Pokes necessary on the C-64.

To read the mouse buttons, use "JOY(1)" with the front port and "JOY(2)" with the rear. If the value returned is 1, the right button was pressed; if it's greater than 127, the left button was pressed.

The RAM Expander

The 6510 or 8510 CPU chip, the brain of the C-64 or C-128 computer, can access only 64K of memory at a time, although Commodore engineers have found clever ways to get around this limit and make more than 64K available to the user. However, the CPU isn't the only chip that needs access to computer memory. Because the VIC chip (which provides all 40-column video support) lacks its own memory, it has to "steal" some temporarily from the CPU. This periodic theft is called direct memory access, or DMA.

While the VIC chip is the only built-in device to use DMA, external devices can access it via pin 13 of the expansion (cartridge) port. The Z-80 cartridge for the C-64 was the first to take advantage of this capability. The RAM expansion unit (REU) is another.

There are three RAM expanders currently available, all identical except for the amount of memory they contain. The 1700 and 1750, for use with the C-128, have 128K and 512K, respectively; the 1764, designed for the C-64, has 256K. Also, the 1764 is packaged with a replacement power supply, because the regular C-64 power supply is inadequate to support an REU, and you'd run the risk of ruining your power supply each time you turned on your computer.

When writing programs for the C-64, don't assume that the user is limited to 256K, since a C-128 owner can still use a 1700 or 1750 expander in 64 mode! Even when writing programs for the C-128, you can't be sure of the amount of REU memory available. There's a "trace" on the circuit board of the 1700/1750 that determines how

much RAM the device has (similar to the one in the 1541 disk drive that determines the drive device number). Cutting the trace tells the REU controller it has 512K.

When the 1700 was originally released, a random five units, with their traces cut, were tested. Three of the five became 1750s! The other two had 256K (in effect becoming 1764s). These changes took place, not because the trace was cut, but because the additional RAM had been installed in the factory. However, there's no guarantee that currently available 1700s will change when you cut the trace, and doing so will certainly void your warranty.

The RAM expander obeys four commands: Fetch (Load), Stash (Save), Swap and Verify. Since these commands are similar to those used on a disk drive, the REU is often called a RAM disk. Unfortunately, you can't use one as a disk drive without special software, because it can't be operated from Basic (at least on the C-64).

The REU works by adding its own input/output registers to computer memory at 57088 ($DF00), and information such as memory size, start addresses, and so forth, must be poked into the proper registers before a final poke is made to the command register to trigger a transfer. While it's possible to do this from Basic, it's not easy; Basic isn't designed for bit-wise operations (where you set or clear a single bit in a byte). Also, you can't access the RAM beneath the ROMs or the input/output (I/O) chips from Basic.

The most effective way to use an REU is, again, through machine language. The 64 RAM EXPAND program pokes a short machine language routine into the cassette buffer, giving you RAM expander access to all the RAM in the C-64—including that under the ROMs and the I/O chips. Use the machine language with the following:

```
SYS 820,A,B,C,D,E
```

Variable A is the computer starting address; B is the computer ending address, plus 1; C is the REU start address; D is the REU bank number (0–7) and E is the command. The commands include 0 for Save (computer to REU), 1 for Load (REU to computer) and 2 for Swap.

The REU "bank" may seem strange if you're a C-64 user. The REU is similar to the C-128 in that its memory is divided into separate blocks, each 64K in length. The 1700 has two banks (numbered 0 and 1), the 1764 has four banks (0–3) and the 1750 has eight banks (0–7).

Note that the bank setting is for the *starting* REU bank and the *starting* REU address. If, during a transfer, the end of a bank is

reached, the transfer will continue, using the start of the next highest bank of REU memory. If the end of the last bank is reached, it will wrap around to the start of bank 0.

If you're a C-64 user, *don't* put any valuable information at 65280 ($FF00) under the Kernal, since the Kernal is used to trigger the transfer. Also, since the C-64 REU routine halts all interrupts, including the nonmaskable interrupt (NMI), using the REU command with a modem or other RS-232 device is not recommended, since it may interfere with data transmission.

It's easier to program the REU on the C-64 than on the C-128—for a couple of reasons. First, the memory of the 128 is more complex. Second, and more importantly, the Basic 7.0 commands supplied to operate the REU may not work properly! Look at the bottom of the last page of the 1700/1750 owner's manual and you'll see reference to a problem in version 0 of the C-128 ROMs, as well as a brief test to see if you have version 0 or version 1 ROMs (if PEEK(65408) returns a 0, then you have version 0).

The brevity of the note implies a minor problem, but it's really far from minor. The "bug" lives in the version 0 Kernal (which means the problem exists even in machine language), and it guarantees that some portion of transfers using any bank but 15 will result in ROM or I/O bytes being transferred instead of the RAM you intended. A second, relatively minor, bug prevents the Bank command from transferring bank 1 RAM. Instead, you have to manually set the VIC chip bank pointer to $D506, so both the VIC chip and the REU see the correct RAM.

Both bugs have been eliminated in version 1 ROMs. If you have version 0, use the program in RAM FIX to effect the fix. This program uses memory at the top of the function key page, but this should be all right unless you use a lot of long key definitions.

Another oddity with the version 0 routine is that it deliberately makes sure the I/O chips are always banked in during the transfer (no matter what you set with the computer's Bank command). This, too, is fixed in version 1 ROMs and in Listing 2.

Basic 7.0 has three commands for RAM expander operation:

```
FETCH A,B,C,D
STASH A,B,C,D
SWAP A,B,C,D
```

Variable A contains the number of bytes to transfer (to a maximum of 65535), B is the starting address in the computer's memory, C is

the starting address in the REU and D is the REU bank number (0–7). In version 1, or version 0 with the fix in place, use the Bank command to determine which computer bank is used for the transfer.

There are two types of interrupts possible on the 128, and version 1 ROMs and RAM FIX take care of only one. The other, the non-maskable interrupt, can't be stopped with a simple SEI command in machine language. Fortunately, the only time you're likely to encounter an NMI (other than pressing the restore key) is when you're using the RS-232 port for telecommunications. In that case, you should restrict yourself to using bank 15 memory with the REU, to make sure the transfer won't be corrupted.

One final word of warning: You *must* have your C-128 running in 1-MHz (Slow) mode before triggering a transfer. Like the VIC chip, a RAM expander can't operate properly at the 128's faster speed.

RUN it right: **C-64**

Trace the Light Fantastic

By Charles Orcutt

COLORFUL SHAPES UNDULATE on the screen, one flowing into the other, fascinating to watch. That's the power of Rainbow Run, my C-64 kinetic art program.

After you run RAINBOW RUN, you'll see a moving straight line segment continually generating curved forms. I've specified the coordinates of the end points of the first line, but then the program takes over, calculating the coordinates of new points and drawing a line between each successive pair.

This process continues indefinitely, but very shortly the program also starts erasing lines, beginning with the first. As pixels are drawn, they are recorded in a memory table, and the location in the table preceding the current position contains the information with which to erase previously lighted points.

The program follows some rules in its creative process. First, it calculates a mirror equivalent for each point, then acts on these points similarly. The mirroring is done from left to right and from

top to bottom, but the sequence can be switched using a joystick in port 2. To activate vertical symmetry, move the joystick left; for horizontal symmetry, move the joystick right. A second rule dictates that when point generation (or what appears as a moving point) reaches a screen boundary, its direction reverses, its speed changes (randomly) and the display assumes a new color.

Rainbow Run is a machine language program in the form of a hex loader, and, although it's only about 1500 bytes long, it occupies all the C-64's available memory. It was inspired by Swish, the impressive kinetic art program by Glen Bredon, but it introduces two important variations.

First, it operates in Hi-Res mode, which provides better horizontal resolution than the Multicolor mode Swish uses. Second, because hi-res calls for more speed, Rainbow Run employs a different technique for providing data for the erasure tail. Swish records the end points of the lines in its memory table, which means the line-draw routine must continuously recalculate the points. Rainbow Run records the memory locations and bit set of the plots, so three bytes are needed to record the erasure tail for each lighted pixel.

RUN it right: **C-64; printer**

A Love-ly Idea

By Jim Borden

FEBRUARY BRINGS OUT the romantic in the stodgiest of us, and sends us in droves to the Hallmark store. But wouldn't your thoughts seem more heartfelt if they were "homemade?" Well, with Valentine Maker, you can design your own greetings, even if you're not handy with pen and brush, then save your designs to disk and print them out for cards. The program works with the Commodore MPS-803 printer and compatibles, as well as most Star and Epson models.

When you run Valentine Maker, the help screen appears and asks you to select a printer. If you have a Commodore or compatible, enter 0. If you have a Star (Gemini) or Epson or compatible, enter 1 or 2. You must specify a printer at this point; you can't do it later.

After you type the printer number and press return, there's a short delay while several machine language routines are read into memory.

The graphic screen and the help screen are Valentine Maker's two displays. The former is for creating designs, while the latter lists the commands you can use and is the launching point for screen saves and loads. You toggle between the screens with control/H.

As the help screen indicates, the control key is used to execute all functions except loads and saves. The functions listed at the top of the help display can be activated only from the graphic screen.

Creating Designs

Your designs can consist of keyboard characters in three sizes and pictures that I've built into the program. The three character sizes are normal (one-column), selected with control/N; big (four-column), selected with control/B; and jumbo (8-column), selected with control/J. Only uppercase/graphics characters are available. You can display them in reverse by pressing control/9.

A special graphic screen cursor, consisting of two dots, indicates the upper-left and lower-right corners of the block that will contain the next character you type. You can see the block if you activate Reverse mode. Since you can't see the block in Regular mode, you can "fine tune" the position of a big or jumbo character by moving the cursor in normal size (where the dots more nearly indicate the size of the character), then switching to the larger size to type the character.

Typing a character at the end of a line moves the cursor to the next line, except when you're at the bottom of the screen. In that case, the cursor advances to the home position.

I've built three pictures into Valentine Maker: a box of candy (control/C or the stop key), a heart (control/L, for love) and a vase of flowers (control/F). The pictures must fit entirely on the screen or their commands will produce nothing. Also, if two overlap, the more recent one will obliterate part of the earlier one.

You can change the pictures to adapt the program to Christmas or any other occasion. Just note that the Data statement above each picture is its printed (not displayed) width and height, and that you can't include any reverse (RVS) codes in the Print statement. Even if you change the pictures, control/C, control/L and control/F will still be the commands to display them.

All the keys except Commodore/shift, the color keys and the function keys work on the graphic screen.

Saving and Loading

You can save your design screens to disk by moving to the help screen and pressing the S key (not control/S). When prompted, type a filename for the screen and press return. If you've already used that name on the disk, an error message will appear and the screen won't be saved. To identify screen files in the directory, Valentine Maker automatically prefixes their filenames with a period. During the save, the display toggles to the graphic screen, then, when the save is complete, the help screen is restored.

Loading a screen is somewhat similar to saving one. Press L, enter the filename (without the period prefix) and press return. The screen will toggle, the new screen will load and the help screen will reappear.

Printouts

You can print a screen at any time by pressing control/P. The screen is dumped as a text/graphics printout, with quotation marks printed as bitmapped characters to avoid Quote mode problems.

With either a Commodore or compatible printer, the picture appears wider in the printout than on the screen. If you have a Star (Gemini) or Epson printer, the dump produces a printout with proportions similar to those on the screen.

You might have to change the secondary address in the Open statement (line 1280) for your interface. If so, use a number that produces Transparent mode without line feeds. A value of 5 works for the Cardco B interface.

Try a few quick pictures to find out if your printer works properly with Valentine Maker; then, if necessary, change the secondary address and try again. When you find the best secondary address, save the program with the change.

If you use a cartridge such as the Epyx FastLoad, you might have to remove it to run this program. Again, do a quick picture, then save, load and print it. If the program doesn't work with the cartridge plugged in, you'll know before you end up with a masterpiece on the screen and a locked-up computer.

RUN it right: **C-64; joystick**

Multicolor Editor 64

By Thomas Teske

MULTICOLOR EDITOR 64 (MC-Editor, for short) is a clever utility that lets you redefine the C-64 character set, save the new sets to disk and use them to build screen displays. You can also save the screen displays to disk and load them sequentially to create very effective animation in your own Basic programs. As you come up with ideas for new characters while designing screens, you can easily create them and incorporate them into your screen displays.

Load MC-Editor using Menu 64, and then enter SYS 16384 if you want to see a program demo—an animated scene of buildings under construction. To use MC-Editor, either enter SYS 16387 (instead of SYS 16384), or press the Commodore key if you're already in the demo.

A series of prompts then appears. At the first, LOAD FROM THE ROM SET Y/N?, enter Y, since you're just starting to use the program and haven't yet saved any other character set. At the second, NAME OF NEW CHARACTER SET?, enter a filename up to 16 characters long. Answer N to the third prompt, LOAD PREVIOUS SCREEN Y/N?, since at this point you have no previous screen to load. Finally, enter a filename at NAME OF SCREEN TO SAVE? For easy identification, I'd suggest keeping these filenames simple, such as SCR1, SCR2, and so on. You must press the return key after entering your response to each prompt.

Once you've answered all the prompts, the program will go into Design mode and display a screen that includes a blank work area at the top and a character set at the bottom. The solidly colored characters at the end of the last row of the character set show the colors you can work with at present.

The only way you can change these colors is by pressing the 1, 2, 3 and 4 keys. These keys have the following effects in memory: 1 changes the background color at location 53281; 2 changes multicolor at location 53282; 3 changes multicolor at location 53283; and 4 changes multicolor RAM at location 55296.

Near the left edge of the Design mode screen you'll see a short

vertical line next to a small square. These are used to access editing modes, as I'll explain shortly.

You operate in Design mode by using an arrow that's controlled with a joystick in port 2. You transfer a character from the character set to the work area by moving the arrow over the character you want, pressing the firebutton, then moving the arrow into the work area and pressing the firebutton again. To avoid a crowded effect, you cannot move a character into the line directly above the character set.

MC-Editor provides two edit modes, one for defining individual characters and one for defining blocks of six characters.

Single-Character Edit Mode

To enter Single-Character Edit mode, move the Design mode arrow over the short line at the left edge of the screen and press the firebutton. The screen border flashes and a "gong" sounds to signify the change in mode. Now move the arrow over a character in the character set and press the firebutton again. A large replica of the four × eight-pixel character appears in a window in the top-left portion of the work area. Back in the character set, a rectangular marker then outlines the chosen character.

You redefine the character by changing the colors of the expanded pixels in the replica, selecting from the preset colors. While holding down the firebutton, press the key of the color you want to use, and move the cursor over the pixels you want to change.

Notice that any changes also appear in the character set below. You can invert the character by pressing I and flip it over horizontally by pressing M (for mirror image).

To redefine another character, press the Commodore key to return to Design mode, re-enter Single Character mode and repeat the definition process. When you've redefined all the characters you want, return to Design mode, where you can save your character set by pressing the C key. If you want to make further changes to that character, you must reboot the program, load in that character set and save it under a different filename.

Large Edit Mode

Now try Six Character, or Large, Edit mode by placing the Design-mode arrow on the square near the left edge of the screen and pressing the firebutton. Once again, the border will flash and a gong will sound. This time, when you move the arrow over a character in the character set and press the firebutton, a three-by-two block

of six large characters will appear in a window that fills about two-thirds of the work area.

The top-left character in the Large Edit window is the one you selected by placing the arrow over it in the character set, where it is now outlined in white. The other five are those in the set that lie to the right and below the selected character. Because of its size, this six-character block may cover part of the screen you're designing, but the covered area will reappear when you return to Design mode.

In Large mode, you can make characters all one color by pressing a key from 1–4, then the firebutton to activate the color, and finally the ← key. You can also move other characters in the character set into the window for editing by "pushing" the cursor against the inside of the window frame.

For example, if you want the two characters in the column to the right of your selected block to go into the editing window, move the work-area cursor right until it hits the right border of the window. As you push the cursor against the border, the other characters you want to edit will move into the window. The white box in the character set will also move, indicating what letter is now upper-left in the edit window.

You can "push" the cursor against the window border in this way in any direction, thus bringing any set of six characters you want into the edit window.

KEYBOARD COMMANDS

F1—Clears either edit window without affecting the character set.

F3—In Design mode, moves a block of characters to another location in the character set. Press F3, move the arrow to the top-left corner of the block and press the firebutton; then move the arrow to the bottom-right corner of the block and press the firebutton again; and, finally, move to the top-left of the new location and press the firebutton a third time. In this and other move procedures, every time you press the firebutton, the screen border flashes and the gong sounds.

F5—In Design mode, moves a single character into the work area. Press F5, move the arrow to the character and press the firebutton; then move to the new location and press the firebutton again.

F7—In Design mode, moves a block from the character set to the screen or from one part of the screen to another. The sequence is the same as for a block move within the set, except you press F7 instead of F3.

B—In Design mode, stores the current screen display in a buffer, then clears the screen. If you press B accidentally, press **T** to restore the screen. The buffer can hold up to four screens at once.

C—In Design mode, saves the current character set to disk.

H—Displays two help screens in Design mode. Press H once for the first screen, twice for the second screen and a third time to return to designing.

I—Inverts the character in the Single Character edit window or a block in the character set in Large Edit mode. Use the same sequence as for other block moves.

M—Mirrors (flips left to right) the character in the Single Character edit window or a block in the character set in Large Edit mode. Use the same sequence as for other block moves. You can create interesting symmetrical designs by using the M and I block moves together.

R—In Design mode, returns the colors to the original setting when you're experimenting with colors or when you want to load a previously saved screen. Approximately 4K, starting at memory location 32768, is configured as a buffer to hold four screens that can be returned with the R command.

S—In Design mode, saves the current screen display to disk. (You can load a screen only at the beginning of the program.)

U—In Design mode, moves the character set to the top of the screen, so you can work at the bottom. **D** moves it down again.

V—Toggles the character set off and on the screen in Design mode. Also, pressing V twice after a save returns the character set to the screen.

↑—In Design mode, smooth-scrolls the top 16 lines of the screen to the left for two screens. This command is great for creating sky scenes.

←—Used with a color key, this paints a character a solid color in Large Edit mode.

+ and −—Change the arrow and rectangular cursor speeds.

Note that once you've initiated any MC-Editor command, it must be completed, since no Escape option exists.

Other Pointers

When you're moving a block from the character set to a new screen location, make certain there's room for it at its intended destination. In other words, ensure that the new location isn't too close to an edge of the character set or to the color characters.

When you edit the space character (the first character in the second row of the character set), your changes will be reflected in all blank areas of the screen. You can create some neat effects this way, especially if you change the space character one pixel at a time. To return the screen to normal, press the 1 and left-arrow keys.

You can erase screen areas in Design mode by placing the arrow in the blank space to the left of the character set and pressing the firebutton, then moving to the work area and pressing the firebutton again on the areas you want cleared. It's possible to erase large areas faster by using F7 to copy a block from the blank area to the left of the character set. ■

If you have questions about Multicolor Editor, please write or call Tom Teske directly at 2878 Redfield St., Niles, MI 49120; telephone 616-683-9255.

EDITOR-IN-CHIEF
DENNIS BRISSON

TECHNICAL MANAGER
LOU WALLACE

TECHNICAL EDITOR
TIM WALSH

MANAGING EDITOR
SWAIN PRATT

SENIOR EDITOR
BETH S. JALA

ASSOCIATE EDITOR
HAROLD R. BJORNSEN

COPY EDITOR
PEG LEPAGE

ART DIRECTOR
HOWARD G. HAPP

DESIGN AND LAYOUT
ANN DILLON

TYPESETTING
DEBRA DAVIES
KEN SUTCLIFFE

FULFILLMENT CONSULTANT
DEBBIE BOURGAULT

10 Programs Included on this Disk

Calendar Program ► Multi-Tasking for the C-64
House Plant Program ► Video Screen Programming
Name and Address Program ► Checkers
Mouse and RAM Program ► Screen Designs Program
Valentines Maker ► Multi-Color Editor

From the January RUN:

- ► 2001: A Calendar Program
- ► Multitasking Comes to the C-64
- ► Raising Rainbows
- ► Screen Basic

From the February RUN:

- ► Address, Please?
- ► Chummy Checkers
- ► Mouse and RAM Expander
- ► Trace the Light Fantastic
- ► A Love-ly Idea

Plus: Extra Bonus Program!

- ► Multicolor Editor 64

If any manufacturing defect becomes apparent, the defective disk will be replaced free of charge if returned by prepaid mail within 30 days of purchase. Send it, with a letter specifying the defect, to:

ReRUN • 80 Elm Street • Peterborough, NH 03458

Replacements will not be made if the disk has been altered, repaired or misused through negligence, or if it shows signs of excessive wear or is damaged by equipment.

The programs in ReRUN are taken directly from listings prepared to accompany articles in *RUN* magazine. They will not run under all system configurations. Use the RUN It Right information included with each article as your guide.